Amendments to the specification,
Marked version of the replacement paragraph(s)/section(s), pursuant to 37 CFR 1.121(b)(1)(ii):

Please replace paragraph [0014] with the following rewritten paragraph:

Among the prior approaches that have been utilized for defining re-usable aspects of program behavior is aspect-oriented programming (AOP). With aspect-oriented programming external metadata or new language syntax is used to define re-usable aspects of program behavior, with "pointcuts" designating locations in existing code which are to be modified, and "advice" which specifies how to modify (i.e., alter, extend, or wrap) the designated code. Aspect-oriented programming is based on the idea that systems are better programmed by specifying the various concerns (properties or areas of interest) of a system and some description of their relationships and then relying on mechanisms in the underlying aspect-oriented environment to compose them together into a coherent program. Aspect-oriented software development makes it possible to modularize crosscutting aspects of a system. Like objects, aspects may arise at any stage of the software lifecycle, including requirements specification, design, implementation, and so forth. Common examples of crosscutting aspects are design or architectural constraints, systemic properties or behaviors (e.g., logging and error recovery), and features. For further information on aspect-oriented programming, see e.g., Bollert, K. "On Weaving Aspects", in International Workshop on Aspect Oriented Programming at ECOOP99 (European Conference for Object Oriented Programming, June 1999), the disclosure of which is hereby incorporated by reference. An example of an aspect-oriented programming system is the AspectJ AOP programming system for Java™ available from the Eclipse Foundation (e.g., currently at ~~www.~~eclipse.org). However, with the AspectJ programming system, source code must be available before aspects can be applied, and aspects cannot be applied or inspected at run-time. In addition, aspects can be somewhat difficult to understand and use.

Please replace paragraph [0032] with the following rewritten paragraph:

Bytecode: A virtual machine executes virtual machine low-level code instructions called bytecodes. Both the Microsoft .NET virtual machine and the Sun

Microsystems Java™ virtual machine provide a compiler to transform the respective source program (i.e., a Java™ program or a C# program, respectively) into virtual machine bytecodes.

Please replace paragraph [0034] with the following rewritten paragraph:

EJB: "EJB" refers to Enterprise JavaBeans, a Java™ API developed by Sun Microsystems that defines a component architecture for multi-tier client/server systems. For further information on Enterprise JavaBeans, see e.g., "Enterprise JavaBeans Specification, version 2.1" available from Sun Microsystems, the disclosure of which is hereby incorporated by reference. A copy of this specification is available via the Internet (e.g., currently at /products/ejb/docs.html made available at java.sun.com~~/products/ejb/docs.html~~).

Please replace paragraph [0036] with the following rewritten paragraph:

Java: Java™ is a general purpose programming language developed by Sun Microsystems. Java™ is an object-oriented language similar to C++, but simplified to eliminate language features that cause common programming errors. Java™ source code files (files with a .java extension) are compiled into a format called bytecode (files with a .class extension), which can then be executed by a Java™ interpreter. Compiled Java™ code can run on most computers because Java™ interpreters and runtime environments, known as Java™ virtual machines (VMs), exist for most operating systems, including UNIX, the Macintosh OS, and Windows. Bytecode can also be converted directly into machine language instructions by a just-in-time (JIT) compiler. Further description of the Java™ Language environment can be found in the technical, trade, and patent literature; see e.g., Gosling, J. et al., "The Java Language Environment: A White Paper", Sun Microsystems Computer Company, October 1995, the disclosure of which is hereby incorporated by reference. For additional information on the Java™ programming language (e.g., version 2), see e.g., "Java 2 SDK, Standard Edition Documentation, version 1.4.2", from Sun Microsystems, the disclosure of which is hereby incorporated by reference. A copy of this documentation is available via the Internet (e.g., currently at

/j2se/1.4.1/docs/index.html made available at java.sun.com~~/j2se/1.4.1/docs/index.html~~).

Please replace paragraph [0042] with the following rewritten paragraph:

XML: XML stands for Extensible Markup Language, a specification developed by the World Wide Web Consortium (W3C). XML is a pared-down version of the Standard Generalized Markup Language (SGML), a system for organizing and tagging elements of a document. XML is designed especially for Web documents. It allows designers to create their own customized tags, enabling the definition, transmission, validation, and interpretation of data between applications and between organizations. For further description of XML, see e.g., "Extensible Markup Language (XML) 1.0", (2nd Edition, October 6, 2000) a recommended specification from the W3C, the disclosure of which is hereby incorporated by reference. A copy of this specification is available via the Internet (e.g., currently at /TR/REC-xml made available at ~~www.~~w3.org ~~/TR/REC-xml~~).